#define LED\_PIN 13 // LED pin for indicating state

#define SENSOR\_THRESHOLD 500 // Threshold for sensor reading

const int sensor0 = A0; // Front sensor

const int sensor1 = A1; // Left sensor

const int sensor2 = A2; // Center sensor

const int sensor3 = A3; // Right sensor

const int enA = 9; // Enable A for motor driver

const int in1 = 8; // Input 1 for motor driver

const int in2 = 7; // Input 2 for motor driver

const int in3 = 4; // Input 3 for motor driver

const int in4 = 5; // Input 4 for motor driver

const int enB = 3; // Enable B for motor driver

int turnCount = 0; // Keeps track of turns at an intersection

bool atDeadEnd = false; // Flag to indicate a dead end

void setup() {

pinMode(LED\_PIN, OUTPUT);

pinMode(sensor0, INPUT);

pinMode(sensor1, INPUT);

pinMode(sensor2, INPUT);

pinMode(sensor3, INPUT);

pinMode(enA, OUTPUT);

pinMode(in1, OUTPUT);

pinMode(in2, OUTPUT);

pinMode(in3, OUTPUT);

pinMode(in4, OUTPUT);

pinMode(enB, OUTPUT);

Serial.begin(9600); // For debugging (optional)

}

void loop() {

readSensors();

checkMazeState();

controlMotors();

delay(10); // Adjust delay if needed

}

void readSensors() {

sensorValues[0] = analogRead(sensor0);

sensorValues[1] = analogRead(sensor1);

sensorValues[2] = analogRead(sensor2);

sensorValues[3] = analogRead(sensor3);

}

void checkMazeState() {

digitalWrite(LED\_PIN, LOW); // Assume not at intersection or dead end

// Check for intersection (all sensors see line except front)

if (sensorValues[0] >= SENSOR\_THRESHOLD && sensorValues[1] < SENSOR\_THRESHOLD && sensorValues[2] < SENSOR\_THRESHOLD && sensorValues[3] < SENSOR\_THRESHOLD) {

digitalWrite(LED\_PIN, HIGH); // Indicate intersection

turnCount = 0; // Reset turn count

atDeadEnd = false; // Not a dead end

}

// Check for dead end (center and both side sensors see line)

if (sensorValues[0] >= SENSOR\_THRESHOLD && sensorValues[1] >= SENSOR\_THRESHOLD && sensorValues[2] >= SENSOR\_THRESHOLD && sensorValues[3] >= SENSOR\_THRESHOLD) {

digitalWrite(LED\_PIN, HIGH); // Indicate dead end

atDeadEnd = true;

}

}

void controlMotors() {

if (!atDeadEnd) {

if (sensorValues[1] < SENSOR\_THRESHOLD) { // Turn left at intersection

turnLeft();

turnCount++;

} else if (sensorValues[3] < SENSOR\_THRESHOLD) { // Turn right at intersection

turnRight();

turnCount++;

} else if (sensorValues[2] < SENSOR\_THRESHOLD) { // Follow line

forward();

} else { // Back up and try other direction at dead end

if (turnCount == 1) {

turnRight();

} else if (turnCount == 2) {

turnLeft();

} else {

stopMotors();

Serial.println("Maze solved!"); // You can add other actions here like playing a victory tune

}

}

}

}

void forward() {

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enA, 255);

analogWrite(enB, 255);

}

void turnRight() {

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enA, 255);

analogWrite(enB, 255);

}```